Feature Selection

BA810: Supervised Machine Learning

Nachiketa Sahoo

Recap

1. Load and explore data, training-test split
2. Consider cleaning and transformations
3. Create pipelines
4. Evaluate various predictive models
5. Finetune hyperparameters of the most promising model (Grid, Random, Bayesian)
6. Estimate error on unused test-data

Course Map (Topics)

1. Introduction to Machine Learning
2. General predictive models
   - Regression and classification
3. Model tuning and selection
4. End-to-end Machine Learning process
5. Specific predictive models
   - Support Vector Machines, Decision Tree, Ensembles
6. Managing class imbalance in practice

Feature Selection

- Why?
  - Improve prediction (models with fewer features are less susceptible to noise)
  - Improve Interpretability
- How?
  1. Best subset selection
  2. Forward/backward stepwise selection
  3. Simpler strategies

- Quick detour to approximate estimation of test error …

Approximate Estimation of Test Error

- Why?
  - Cross validation estimates through out of sample prediction, but computationally costly
- How to approximate?
  - Adjust in-sample error (obtained for free during training) for number of parameters
- For linear regression
  - $C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$, same as another general criterion AIC
    - d: number of features, $\hat{\sigma}$: estimated standard deviation of the error
  - $BIC = \frac{1}{n}(RSS + \log(n)\, d\hat{\sigma}^2)$,
  - Adjusted $R^2 = 1 - \frac{\frac{RSS}{n-d-1}}{\frac{TSS}{n-1}}$, contrast to $R^2 = 1 - \frac{RSS}{TSS}$
    - Where $RSS = \sum_i (y_i - \hat{y})^2$ and $TSS = \sum_i (y_i - \bar{y})^2$

Best Subset Selection

Evaluate all possible models using the d features.

1. Start with the null model (M_0 : predicts mean of y)
2. For $k = 1, 2, \ldots, d$:
    1. Fit all possible models that use exactly k predictors
    2. Pick the best model per R^2 (or cross validation) and call it M_k
3. Select the best model from $M_0, \ldots, M_d$ per either cross validation error, C_p, AIC, BIC or adjusted R^2

Forward Stepwise Selection

Add the best next feature given the features included already.

1. Start with the null model (predicts mean of y)
2. For $k = 0, 1, \dots, d-1$:
   1. Fit $d - k$ possible models that add only 1 predictors
   2. Pick the best model per R^2 (or cross validation) and call it M_k
3. Select the best model from $M_0, \dots, M_d$ per either cross validation error, C_p, AIC, BIC or adjusted R^2

Backward Stepwise Selection

Remove the next least contributing feature given the rest.

1. Start with the full model (with all features in it)
2. For $k = d, d-1, \dots, 1$:
    1. Fit k possible models each removing a separate predictor
    2. Pick the best model per R^2 (or cross validation) and call it M_k
3. Select the best model from $M_0, \dots, M_d$ per either cross validation error, C_p, AIC, BIC or adjusted R^2

Comparisons

- Best subset
  - Finds the best of all possible subset of features
  - Slow, can overfit ("multiple comparison problem" when comparing large number of models)
- Forward stepwise search
  - Fast, less likely to overfitting, can be used for linear regression with $d > n$
  - Need not find the best subset, can miss feature groups that complement
- Backward stepwise search
  - Fast, less likely to overfitting, **cannot** be used for linear regression with $d > n$
  - Need not find the best subset, but **less likely to miss pairs that complement**

Simpler Strategies

For when we have too many (>50) features

- Eliminate hopeless features
  - >50% missing
  - All have same value (VarianceThreshold)
- Assess each feature independently
  - Reduces the number of cases to evaluate
  - Select those most related to outcomes
    - F-statistic, mutual information, chi-square, …
- Important features as given by a model
  - Importance: magnitude of coefficient in linear models, or gain measure in decision tree
    - For linear model choose methods such as Lasso that promote turn features off (promote sparsity)
  - Fit once, pick the top K features
    - Could tune K by grid search

- Recursive Feature Elimination
  - Builds on model-based selection

  1. Estimate the model with all features, remove the least "important"
  2. Repeat until desired number of features remain

  Alt:

    - Measure CV error after step 1.
    - Eliminate features until one remains.
    - Pick the number of features (and corresponding features) that leads to the smallest CV error.

These get progressively slower.

Summary

- Weed out the hopeless features
- If sequential feature selection is computationally feasible, go ahead
- Else, see the best among the simpler strategy that is feasible as an intermediate pass

Announcements

- Provide team feedback by tomorrow (11/16) noon
- Next class: support vector machines
  - Andrew Ng's course video on SVM
  - No class on (11/22) next Wednesday (Thanksgiving)
- More Datacamp chapters post break
  - Plan ahead